BGP Secure Routing Extension

– BGP-SRx –

Version 0.3

User's Manual

March 2013

www-x.antd.nist.gov/bgpsrx

bgpsrx-dev@nist.gov

Oliver Borchert, Kyehwan Lee, Kotikalapudi Sriram, Doug Montgomery

Index

Introduction

BGP SRx was developed by and continues as an ongoing project in the Advanced Network Technology Division (ANTD) at the National Institute of Standards and Technology (NIST). This effort is supported by the Department of Homeland Security under the Secure Protocols for the Routing Infrastructure (SPRI) program and the NIST Information Technology Laboratory Cyber and Network Security Program.

The BGP SRx provides an API to router implementations that wish to use origin validation as well as path validation. (Note: Path validation is not yet implemented but will be incorporated into SRx in the near future). The BGP-SRx is meant to be used as a validation service. This keeps the impact (software/memory/processing) on the router to a minimum. SRx provides an API that allows embedding a small proxy into the router. This proxy handles the router proxy communication.
This design allows the SRx-Server to serve multiple BGP router instances.
<u>What BGPSrx is not:</u> BGPSRx is not an RPKI validation cache. It does not rsync with RPKI repositories nor does is validate certificates. SRx-Server synchronizes with RPKI validation caches using the RPKI to router protocol as described in RFC 6810.
At this point, the SRx-server uses plain TCP as transport for communication with the RPKI cache(s).

Status of BGP-SRx

The current version of BGP-SRx is NOT intended to be used in a production system even though we attempt to get it to production level. BGP-SRx is still in the stage of a prototype and might face some instability. In such a case we appreciate every input that helps us to improve the stability as well as performance. The code is open source. Feedback can be send to bgpsrx-dev@nist.gov.

<u>What is new?</u>

The new SRx Server 0.3.0 is faster than its predecessor. Multi-client handling is implemented and the communication between the SRx proxy API and the SRx server does not use "broadcasts" anymore. Internal bugs in the validation requests when no ROA's are installed have been removed, and the communication between the SRx Server and the Proxy-API has been overhauled. The protocol between SRx Proxy API and SRx server changed dramatically. In version 0.2.0 of the protocol, the proxy was required to wait until an answer from the server was received. Version 0.3.0 of the proxy/SRx server communication protocol does not do that anymore. The client provides a local ID with each new validation request and the server sends a notification back, providing the local ID as well as a system wide unique update id. This allows the client (router) to identify the update to assign the unique update ID provided by the SRx server. The change was necessary considering the processing spike of updates during table dumps with 70+K updates. In such cases, the performance started to degrade to unacceptable levels. The new version allows the router to keep going without the necessity to wait for the validation result. Also the possibility of update ID collisions forced change in this procedure. Previously in version 0.2.0 of the implementation, the SRx proxy calculated the update ID in case of request timeouts. This posed problems especially if the newly calculated update ID collided with an ID already assigned to a different update. The calculated update ID is a modified version of CRC32 and collisions are expected and therefore should only be generated at the SRx server. This is how it is done in version 0.3.0. An additional major change is in the way the BGP updates are managed in the SRx server. Updates are now mapped to the proxy client, which removed the need of broadcasting the result to all clients, even the once that do not hold the update. In addition this allows a better memory management on SRx Server side. Also the capability of handling multiple clients is added.

Important Changes:
The SRx server changed verification state labels described in rfc6482 "Valid", "Unknown", and "Invalid" into the labels described in rfc6811 "Valid", "NotFound", and "Invalid". Furthermore the SRx-server does not accept ROA information for any AS number specified in rfc5368. Updates using any of these AS numbers will be validated though, but can only achieve the origin validation state "NotFound" or "Invalid". This allows the client (router) that uses the proxy to send validation requests for updates where the origin cannot be determined due to path aggregation. In such cases the client just uses an ASN number specified in rfc6811.

SRx Server Performance

The SRx Server performance was tested on Emulab nodes with 12BG RAM, 8 Processor Cores, 2.8GHz. The topology was generated in Emulab with one physical server dedicated to each router. This was done intentionally to be able to see the difference between "on board" SRx Server <-> router communication and remote communication. The tests include SRx server and BGP router located on the same physical platform and as well as on distributed platforms. In each test we used the SRx-server-client test harness and send 1,638,400 unique validation requests with receipt request. Each request provided "undefined" as default origin validation value. This means that in case the update is not yet known to the SRx server, the SRx server first responds with a receipt notification where it populates the update ID plus the default value provided by the proxy. This notification is then followed by a validation notification that contains the calculated origin validation result. Because the SRx server validation never results in "undefined" as a validation result we can be assured that the second notification will come, in this experiment scenario with "NotFound". The reason for the "NotFound" result being returned in the notification is that the tests were performed with initially no updates and ROAs stored in the SRx Server.

Test1:
Main client and server are located on the same physical machine. Traffic does not get routed over the "wire". The third column contains the timing results of repeated requests using 2 clients with one remote client.

1st Time Request	Repeated Request	2 Clients (one local, one remote)
58.204 μs	40.675 μs	75.425 μs

Test2:
Client and server are on physical different machines connected using a 100Mb link.

1st Time Request	Repeated Request
48.772 μs	39.593 μs

Based on these tests, we see that remote communication does not have any impact on the validation response time whatsoever. The biggest difference though is in the validation and data generation itself. It appears that the separation of server and client performs better possibly because the SRx server and client software modules do not need to share any local resources such as memory and processor.
With regard to performance gain in scenario of the multiple clients, sending the same BGP updates, we expect to see better numbers when it comes to path validation, when the CPU processing times per update expected to be much larger. Also, we see room for additional improvement by tweaking the parallel processing.

Installation

BGP-SRx provides configuration scripts. To install SRx, download the package from http://www-x.antd.nist.gov/bgpsrx. Once downloaded, deflate the package and call the configuration method. The file labeled "INSTALL" contains an example on how to configure the SRx server. This package also generates the SRx-API that is needed for QuaggaSRx. Follow the configuration messages and install missing libraries as needed. Under Fedora or CentOS, all libraries can be installed using yum. The internal prefix tree (Patricia) tree is bundled in this package. If you decide to use the default one, check the patch file located in SRx/extras and update the installed version. Recompilation will be necessary. If not, use the switch --with-patr, and BGP-SRx will be compiled with the bundled version.
Once compiled, call "make; make install". To select an individual installation directory use the configuration parameter --prefix. This directory is needed for QuaggaSRx to specify the location of the API binaries.

Note: The software is developed and tested primarily on CentOS 6 and Fedora 15

The SRx Server Software [srx_server]

The BGP-SRx Server is the main component that collects ROA information from the RPKI validation cache using the router to cache protocol (RFC 6810). The BGP router connects to SRx using the SRx-API and sends update validation requests for origin validation and path validation. The SRx server answers by sending the validation results either in two separate notifications, one for origin validation and one for path validation depending on availability or bundled in one single notification containing both results. The validation results are processed independently from each other as pure origin validation and pure path validation. The path validation process does NOT include origin validation. The combining of these two results for the BGP decision process MUST be done within the router. This allows for implementing different customized strategies in the router and is very helpful for research purposes.
Once the SRx Server recognizes a change in the validation state of either of the validations, it sends the appropriate notification to all routers, registered for the particular update.
The SRx Server can be remotely accessed by using a telnet client. It is recommended to combine telnet with rlwrap to gain command history. Using the alias command allows easy combination of rlwrap with telnet: alias telnet='rlwrap telnet'.
As soon as a telnet session is established (only a single session is supported at this time) a list of all available commands can be retrieved using the *'help'* command.

Available commands are:

close, quit, exit	Close this console!
shutdown <password>	Shutdown the SRx Server!
log-level [<number>]	Set or show the log level of the server. If no log level is provided, the server returns the current log level. The following log-levels are supported: 3=ERROR, 4=WARNING, 5=NOTICE, 6=INFO, 7=DEBUG
rtr-sync [proxyID]	Send synchronization request to the provided proxy or all. (Currently to all)
num-updates	Display the number of updates stored in the update cache and shadows stored in the prefix cache. Only updates in the prefix cache are verified.
num-prefixes	Display the number of prefixes stored in the prefix cache!
num-proxies	Display the number of proxies attached.
command-queue	Displays the number of queued commands in the command queue.
show-srxconfig	Display the configuration of the SRx server
show-update <id>	Display update data with the ID (hex or decimal).
show-proxies	Display the proxy mapping.

dump-ucache	Dump the update cache to command line of SRx!
dump-pcache	Dump the prefix cache to command line of SRx!

(WARNING: the dump-xxxx commands dump the complete cache content on the command line. This function should only be used for debugging of small data sets!)

!! [<parameter>]	Repeat last command with optional new parameter if specified, otherwise old parameter!

SRx Configuration Settings:

The SRx Server requires a configuration file which is expected to be either in the directory from where it is called or provided using the parameter –f <conf-file>. Most configuration settings can be handed over as command line parameters. In case of a conflict to between a command line parameter and the corresponding configuration script, the command line parameter has precedence over the configuration script. The server will NOT start without a configuration file.

Configuration File Parameters:

```
# print information on the command console
# verbose  = true|false;
verbose = true

# specifies the log level for output (3=ERROR, 4=WRNING, 5=NOTICE,
# 6=INFO, 7=DEBUG)
#loglevel = 3|4|5|6|7;
loglevel = 3;

# specify the log file name, otherwise log information will be send to the
# console.
log     = "/var/log/srx_server.log";

# if enabled the SRx server will send a synchronization request to the router.
# It is expected that the router will send validation requests for all updates in
# its tables to SRx server.
# sync = true|false;
sync    = true;

# The port address the SRx server is listening on for connections from the
# router.
port = 17900;
```

```
# Console configuration:
console: {
        # Port address of the server console.
        port = 17901;
        # The password used to shutdown the BGP-SRx server
        password = "x";
};

# Configuration for Validation Cache:
rpki: {
        # Server address of the validation cache
        host = "localhost";
        # Port address of the validation cache. Protocol: router to cache
        port = 50001;
};

# Configuration for BGPSEC data server: This is one of the hooks for future
# BGPSEC validation. The setting is not used yet but still mandatory!
bgpsec: {
        # Server address of the signature cache
        host = "localhost";
        # Server port of the signature cache
        port = 50002;
};

# These experimental settings are used to manipulate the internal execution
# flow of the server.
mode: {
        # Turn off the send queue (true|false)
        no-sendqueue = true;
        # turns off the receiver queue (true|false)
        no-receivequeue = false;
};

# It is possible to pre-configure the internal proxy-client mapping. This
# mapping is used to identify which client is linked to what update. It is
# possible to configure up to max 255 clients.
mapping: {
        # client_x = y  with x = 1..255 and y either an IPv4 or 4 byte integer
        client_1        = "2";
        client_10       = "10.0.0.1";
};
```

IMPORTANT: Command line parameters overrule configuration script parameters.

Configuration as command line parameter

-h, --help	Display this help and exit
-f <file>, --file <file>	Specify a configuration file
--credits	Displays the developers information
--version	Displays the version number
--full-version	Displays the full version number
-v, --verbose	Enable verbose output
--loglevel <level>	The log level for the verbose output. The following levels are supported: (3)=ERROR, (4)=WANRING, (5)=NOTICE, (6)=INFO, (7)=DEBUG
-l <file>, --log <file>	Write all messages to a file
--syslog	Send all messages to syslog
-C <#>, --proxy-clients <#>	Minimum expected number of proxy clients. By default this value is 2. It affects the internal memory consumption / performance per update.
-s --sync	Send synchronization request each time a proxy connection is established!
-k --keep-window <sec>	The default keepWindow in seconds. Zero deactivates this feature.
-p, --port <#>	Specify the listening port (def.: 17900)
-c, --console.port <#>	Specify the console port (def.: 17901)
-P <pwd>, --console.password <pwd>	Password for remote shutdown
--rpki.host <name/ip>	RPKI/Router protocol server host name
--rpki.port <#>	RPKI/Router protocol server port number
--bgpsec.host <name/ip>	BGPSec/Router protocol server host name
--bgpsec.port <#>	BGPSec/Router protocol server port number
--mapping.client_# <ip/#>	A pre-defined proxy/client mapping. The client number ranges from 1..255 the proxy id can be given as either IPv4 or 4 byte integer

Experimental Options:

--mode.no-sendqueue	Disable send queue for immediate results.
--mode.no-receivequeue	Disable the receive queue. This queue allows to push the processing of packets into its own thread.

Tools

This software package comes with a set of tools used to test SRx-server and its components. These tools simulate a validation cache, a validation cache client, and a router / SRx-client.

srxsvr_client

This tool is an example implementation of the BGP-SRx Server Client. It helps to test functions of the SRx server without the need of a full-blown API implementation such as QuaggaSRx. In addition this test harness also provides a statistics framework that allows measurement of the performance of the SRx Server from its client's perspective.
To operate this test harness, a command line console is provided, with a set of commands to connect/disconnect to a BGP-SRx server, to send validation requests, etc. It implements most of the API and can be used as an example implementation for someone who wants to use the BGP-SRx API.
The Help command provides a list of commands this client can send. Also most commands with parameters can be used with default values. For example the connect command, entered with an incomplete set of attributes will request the necessary parameters and provides default values. In addition, this tool allows code completion using the tabulator key.

Console commands:

Each command can be called without providing parameters. Missing parameters will be requested.

LOG_LEVEL <#>
    set the log level of this test harness.
    (3)=ERROR, (4)=WANRING, (5)=NOTICE, (6)=INFO, (7)=DEBUG

RESET_PROXY <ip/#>
    Create a new proxy instance and set its default proxy ID.

NON_BLOCKING_SOCKET <true|false>
    As long as the proxy is not connected the socket type can be changed. Changing the socket type also changed the operational mode of the SRx-API itself. Only an external controlled proxy provides a non-blocking socket.

connect <host> <port> <proxy-id> <peer-as> [<peer-as>*] <0>
    Connect to the SRx-server using the provided information.

disconnect
    Disconnect from the SRx-server

reconnect
    Disconnects and re-connects.

addPeer <asn> [<asn>*] <0>

Add the given peer or peers. The last peer MUST be 0.

delPeer <asn> [<asn>*] <0>

Add the given peer or peers. The last peer MUST be 0.

verify <localID> <method> <asn> <prefix> <def-oval> <def-pval> <string>

Send a verification request to the SRx server.

localID:	an ID > 0
method:	0=just store, 1=ROA only, 2=BGPSec only, 3=both
asn:	The origin AS
prefix:	The prefix information
def-oval:	The default origin validation result 0=valid, 1=unknown, 2=invalid, 3=undefined
def-pval:	The default path validation result 0=valid, 2=invalid, 3=undefined
string:	Some text string to simulate the remaining BGP update byte string. This parameter might be changed in future versions.

sign <update-id> <prepend-counter> <peer-as>

Accepted but not yet implemented by SRx Server

delete <keepWindow> <update-id>

Request to delete the provided update after keepWindow seconds. The SRx-server will remove the update-client association but depending on memory need or other linking deletes the update at any time or not at all.

run <script-file>

Executes the commands found in the provided script. (Console output is suppressed during script runs – except statistic prints)

stat-init

Initialized the statistics framework

stat-mak <#>

Set a trigger when the statistics are generated. This command sets the trigger for a given number of received notifications.

stat-mark-nr <#>

Set a trigger when the statistics are generated. This command sets the trigger for a given number of received notifications except receipt notifications.

stat-print
Print the statistics. Will be done automatically if the stat-mark or stat-mark-nr trigger is set.

stat-start
Start the statistics framework.

stat-stop
Stop the statistics framework.

exit, quit, \q
Exit the program.

help
Help screen with quick explanation of the commands

rpkirtr_svr

command: prkirtr_svr [port – Def: 50001]
The tool is a Validation Cache simulator. The default port is 50001. It provides the validation cache interface to
BGP-SRx and can be used to inject ROA information to the system. The command line console can be used to add and delete ROA entries. It also allows loading ROA information via a script. Refer to the example files provided in this distribution.
This tool also provides a Help command.

Console Commands

verbose
Toggle to turn verbose mode on and off
cache
Display the content of the validation cache
version
Display the version of the tool.
sessionID
Display the current session id
help [<command>]
Display this screen or detailed help for the given command!
credits
Display credits information!
empty
Empties the cache
sessionID <number>
Generates a new session id.
append <filename>
Appends a prefix file's content to the cache
add <prefix> <maxlen> <as>
Manually add a whitelist entry

addNow <prefix> <maxlen> <as>
    Manually add a whitelist entry without any delay!
remove <index> [end-index]
    Remove one or more cache entries
removeNow <index> [end-index]
    Remove one or more cache entries without any delay!
error <code> <pdu|-> <message|->
    Issues an error report. The pdu contains all real fields comma separated.
notify
    Send a SERIAL NOTIFY to all clients.
reset
    Send a CACHE RESET to all clients.
quit, exit
    Quits the loop and terminates the server
clients
    Lists all clients
run <filename>
    Executes a file line-by-line
sleep <seconds>
    Pauses execution for the given amount of seconds

rpkirtr_client

Command: rpkirtr_client [<server-def:localhost> [<port-def:50001>]]
This tool allows testing a validation cache. In this case it can be used as tester for the rpkirtr_svr tool. It helps to debug the RPKI cache test harness without the need of a full-blown BGP-SRx instance.

Support

Before contacting us, please verify that the SRx Server and its client are connected and properly communicating. We provide tools such as Wireshark plugins that allow analyzing the traffic in a human readable manner. Also check firewall settings. If nothing helps (not even a reboot), please contact us and we will try to help. In case of crashes, please provide a description on how to reproduce the crash and if possible a core dump.
To be informed of bug fixes or ask questions to the community, subscribe to the users email list by sending an email to bgpsrx-users-request@nist.gov with subscribe in the subject.

Questions to the developers and general contact information:

Email:	bgpsrx-dev@nist.gov
Web:	www-x.antd.ist.gov/bgpsrx

Developers:

Oliver Borchert	oliver.borchert@nist.gov
Kyehwan Lee	

Previous Developers:

Patrick Gleichmann (V0.1.0 only)